操作系统系列丛书

Windows NT4
Advanced Programming

Windows NT4
高级程序设计

(美) Raj Rajagopal
Subodh P.Monica 著
前导工作室 译

机械工业出版社
McGraw Hill
CMP

本书主要讲述如何进行 Windows NT 的高级程序设计，结合了很多编程实例，对 Windows NT 4.0 高级程序设计的方方面面作了详细的讨论，并给出了很多很好的、立即可用的例子程序。对于对 Windows NT 编程感兴趣的读者，尤其是有 Windows NT 编程经验的程序员来说，这是一本不可多得的好的参考书。

本书内容新颖详实，叙述深入浅出，结构严谨，适合于大中专院校师生、计算机软件开发专业或非专业人员以及任何对 Windows NT 编程感兴趣的读者。

本书版权登记号：图字：01-98-0620

图书在版编目（CIP）数据

Windows NT 4 高级程序设计/（美）拉加歌帕（Rajagopal，R.），（美）莫尼卡（Monica，S.P.）著；前导工作室译．—北京：机械工业出版社，1998.5
（操作系统系列丛书）
书名原文：Windows NT 4 Advanced Programming
ISBN 7-111-06305-8

Ⅰ.W… Ⅱ.①拉… ②莫… ③前… Ⅲ.窗口软件，Windows NT-程序设计 Ⅳ.TP316

中国版本图书馆 CIP 数据核字（98）第 08196 号

出 版 人：马九荣（北京市百万庄大街 22 号 邮政编码 100037）
责任编辑：江 颖
北京第二外国语学院印刷厂印刷·新华书店北京发行所发行
1998 年 5 月第 1 版第 1 次印刷
787mm×1092mm 1/16·40.75 印张
印数：0001-5000 册
定价：89.00 元

前　言

我们在开发应用程序过程中发现对特定的项目，操作系统的某一部分比其它部分有更多的使用频率。如对面向通信的 Windows NT 项目，将主要使用 Socket，而在面向因特网（Internet）应用中，将大量使用 WinInet 和 ISAPI。我们一直想写一本介绍操作系统各个部分的书，并能给出一些例子，让用户使用这些例子可快速地生成自己的应用程序。本书致力于满足上述要求，适用于使用过 Windows NT 并想学习新的 Windows NT 主题的高级程序员。基于这种考虑，本书涉及了大量的 Windows NT 编程主题，包括 OLE2、ActiveX、WinSock2、ODBC、DAO、WinInet、ISAPI、TAPI、OpenGl、高级控件、音频、视频、3D、动画、GDI 等。每个主题的介绍使得有经验的程序员能很快地了解主题概念和给出的例子，并能在很短的时间内就能开发自己的应用程序。

作为程序员，我们花费了一些时间寻找并使用一些编程辅助工具，本书英文原书的 CD 中包含了许多这样的辅助工具。你可能会发现一两个没用过却很有用的工具。

好了，开始编程吧!!

目　录

第三部分　NT 通信程序设计

第一部分 Windows NT 编程基础

第 1 章 Windows NT 总览

Windows NT 是程序员的乐园，与 UNIX 和 NetWare 一样，它是有竞争力的操作系统（OS）和网络操作系统（NOS），微软做了大量的工作来改进以前 Windows NT 版本中的问题，尽管 Windows 95 仍然有很大的安装基础，但与 Windows NT 的差别正在变小。从 4.0 开始，NT 与 95 有相同的用户界面，Win32 API（Application Programming Interface，应用程序编程接口）成为两者共同的开发机制。从“Cairo”（微软下一版本的代号）开始，NT 将受益于面向对象系统。随着内存条、处理器和其它硬件价格的下跌，运行 NT 的最小系统要求将越来越容易满足。NT 将不再有 16 位代码，并比 95 稳定，它也是微软 BackOffice 产品的基础。尽管 UNIX 仍有许多关键任务的应用，但许多 UNIX 应用程序开发商已经开发了 NT 上的等价的应用程序，同时有第三方的产品使用户能将 UNIX 上的应用程序转到 NT 上。由于具有因特网信息服务（IIS）和访问企业数据的能力，NT 在快速发展的企业 Intranet 中将扮演重要角色。您接受 Windows NT 了吗？本书是为 NT 程序员写的，目的是使他们了解 NT 编程概念。书中附有大量的例子程序，这些代码可用于您的应用程序，随书的 CD 还带有编程辅助工具。

第一部分从程序员的角度给出了 Windows NT 的总览。Windows NT 可分为两类——NT 工作站（Workstation）和 NT 服务器（Server），NT 工作站与服务器的编程大部分相同，因此，除了特别声明，所有的编程主题二者均适用。本章后部将总结 NT 工作站与服务器的相同点与区别。

本章与第一部分其它章的目的是给出 Windows NT 的基本概念，在进入高级编程前您必须熟悉这些概念。如果您已经使用过 NT，则可能熟悉给出的主题中的一部分。由于本书的后续部分假设您已经熟悉这些主题，所示您必须仔细阅读本章并确实掌握列出的主题。若想更详细地了解每个主题，可参阅 Herbert Schildt 编写，由 Osborne/ McGraw-Hill 出版社于 1997 年出版的《Windows NT 4 Programming from the Ground UP》。

微软为进行 Windows NT 编程提供了编程语言、API、SDK（软件开发包）、类库和一个开发环境等。Windows NT 程序员必须掌握或了解的主题包括：

- Windows NT 结构总览
- 编程语言和开发环境
- 应用程序编程接口（API）
- 软件开发工具（SKD）
- 微软基础类库（MFC）
- 图形设备接口（GDI）
- 因特网和网络编程

■构件对象模型（COM）和分布 COM
■对象嵌入与链接（OLE）
■多媒体
■数据访问
■注册库

下面将更详细地介绍每一个主题。

1.1　Windows NT 结构总览

图 1-1 能很好地说明 Windows NT 结构，示出了组成 NT 结构的不同部分、它们的运行模式和构件间的互操作等。

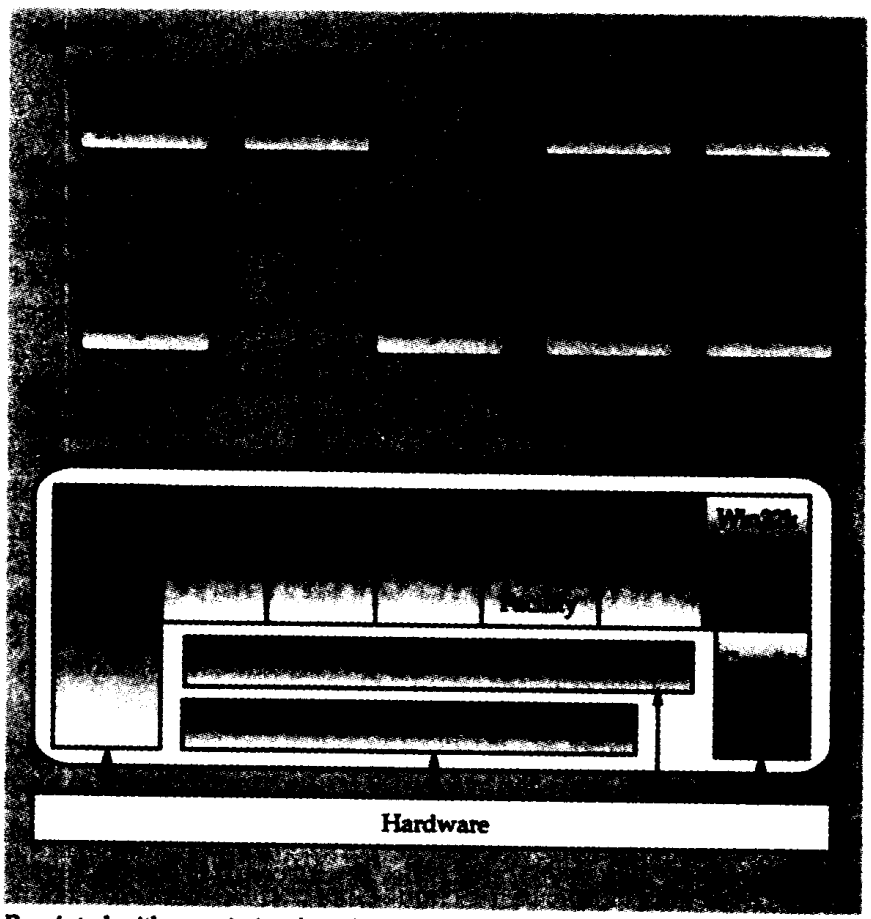

图 1-1　Windows NT 结构总览

1.1.1　内核与微内核

只要计算机打开，总在运行的一个程序是操作系统（OS），它使用真正的主存、磁盘空间和其它资源，这些是使计算机工作的必要的开销。操作系统设计人员的目标是用尽可能少的开销完成 OS 应该完成的任务。NT 减少开销的方法是使基本的操作系统尽可能小而紧凑，只有那些在其它地方不能正常运行的功能才被放于基本操作系统或内核中。NT 的这种基于微内核的方法与 Mach 相似，Mach 是卡内基梅隆大学研制的。

内核是操作系统的核心。在图 1-1 中可看到内核与操作系统其它部分的关系，内核位于

屏蔽硬件的代码层之上，即硬件抽象层（Hardware Abstraction Layer，HAL）之上，本章后面将讨论 HAL。

内核驻留在内存中并且不能被抢先（除了某些中断）。内核的功能有：

- 处理硬件异常和中断
- 调度、赋优先权和分派基本执行单位——线程（thread）
- 在多处理器环境中同步各处理器上操作

操作模式（Operating Modes）

程序员必须记住的一个编程原则是，在您的程序运行期间，该程序与其它程序和 NT 操作系统轮流占用处理器。若程序在多处理器系统中运行，则有多个程序（或甚至同一程序的多个线程）同时在多个处理器上运行，但处理器上程序仍然轮流执行，主存、磁盘空间和其它资源由所有正在执行的程序共享。必须提供一种机制将应用程序与操作系统和其它应用程序分隔开来。Windows95 的应用程序分离不很好，常导致应用程序主存崩溃，产生通用保护错（GPF）。

致 UNIX 程序员

尽管 UNIX 与 NT 在很多方面都相似，但有一个不同点是内核。Windows NT 使用基于微内核技术，内核核心功能尽可能地缩小，而将操作系统的其它功能放在系统的非优先部分——保护子系统（Protected Subsystems）中执行。UNIX 则不同，它的内核要大得多，并包括了许多操作系统功能。大内核的结果是，与 NT 内核相比，常常要做更多的修改。

Windows NT 解决这个问题的方法是使操作系统代码在处理器高优先级上运行，如核心态（kernel mode）。在核心态运行的操作系统代码可访问系统数据和硬件。运行于处理器非高优先级的应用程序，如用户态（User mode）对系统数据和硬件的访问有限制。因此，若某个错误程序非法运行时，操作系统将可以控制并且能终止它，而不影响其它程序。Windows NT 使用处理器（也叫中央处理单元或 CPU）提供的保护机制（也叫环，rings）来实现态分离。当应用程序运行时，操作系统不使用 CPU，也不知道应用程序要做什么事。然而操作系统占用 CPU 来检查该应用程序是否试图在与其不相适应的环级上运行，若应用程序运行不正常，则 CPU 产生一个异常并激活操作系统，这时操作系统控制并处理这个应用程序。

若某应用程序要访问硬件，——如某个应用程序要打印或从磁盘读数据，它激活操作系统服务，通常是通过一组定义好的接口——应用程序接口（Application Programming Interface，API）。

1.1.2 硬件抽象层（HAL）

操作系统，包括 Windows NT，设计适于在多种硬件平台上运行，由硬件抽象层（HAL）将特殊平台的细节屏蔽掉。Windows NT 设计适于在 Intel、Alpha 和 PowerPC 上运行，尽管最近 IBM 和其它厂家宣称不再支持 NT 向 PowerPC 的移植。虽然在指令集（精减指令集 RISC 和非 RISC）、字长（64 位和 32 位）、甚至处理器数目方面硬件都各不相同，但由于有 HAL，操作系统的大部分都不用考虑这些硬件差别。

处理器支持（Processor Support）

直到最近，许多桌面计算机和服务器都是单 CPU 的，大型机则使用多处理器，即在同一机器内有多个处理器。随着处理器的降价和支持多处理器的操作系统的出现，桌面计算机和服务器中使用多处理越来越常见了。

非对称多处理（Asymmetric multiprocessing，ASMP）中，操作系统专门占用一个或多个处理器，并在剩余处理器上调度运行应用程序。支持对称多处理（Symmetric multiprocessing，SMP）的操作系统对处理器就没有这种限制，可在任一个处理器上运行任何程序的能力提供了更好的负载平衡（在 ASMP 中，当应用程序在等待占有处理器时，运行操作系统的处理器可能是空闲的）。SMP 中的容错性也好一些，因为在 ASMP 中当用于运行操作系统的某一处理器有错误时，尽管其它处理器是正常的也意味着计算机不能正常运行。改进负载平衡和容错能力的代价是昂贵的。SMP 操作系统的设计和维护很复杂，Windows NT 和许多 UNIX 操作系统都支持 SMP。对应用程序，SMP 支持是透明的。

1.1.3　执行

在 Windows NT 中，执行（Executive）指的是运行于核心态的操作系统代码。除了内核和硬件抽象层，执行还包括为应用提供存储管理、I/O 处理、对象处理、进程管理、安全监测和本地过程调用等服务的模块。这些模块不是以一个叠加在另一之上的层次形式实现的，而是以对等管理器的形式实现，它们之间有着互操作。

致 UNIX 程序员

与 UNIX 不同，Windows NT 中的这一部分不能由本地系统管理员修改，只能由微软的升级版更新。

进程管理器

进程是程序的运行实例。每个进程有自己的存储地址空间（Windows NT 和大多数 UNIX 系统中为 4Gb），构成进程的代码就驻留于其中；每个进程还拥有运行所要求的文件、线程等资源。执行的进程管理部分负责进程（和线程的）创建、管理（包括暂停和继续进程的执行）和删除。与其它操作系统服务，如安全性引用监测器一样，进程管理集中了有价值的性能数据，以帮助系统管理员监测系统性能。

致 UNIX 程序员

与 UNIX 不同，Windows NT 中的进程不能自己运行，同样，NT 中当一个进程创建另一个进程时，不会自动地建立父子关系。

尽管进程是运行实例，进程本身并不运行。一个进程由一个或多个线程构成，线程是执行的单位。一个进程至少有一个线程，也可创建多个线程，每个线程有其存储空间。线程是一种划分可在后台执行的功能的方便而有效的手段，而主线程可继续其它处理。在多处理器系统中，同一进程的两个或更多个线程可并行执行，这样同一程序的某一部分可并行执行。然而，方便和有效的代价是要同步（Synchronize）线程。若您用一线程在后台做一个耗时的

排序操作，则主线程要确定排序的输入不变，且在排序完成后必须通知主线程。正如将在第 10 章中所说的那样，使用线程是发挥 Windows NT 编程优势的一种方法，您可以使用诸如信号量（semaphore）的通讯技术来同步线程。

存储管理器

前面提到每个进程的地址空间为 4Gb，而许多桌面计算机和服务器没有这么多的实际主存。考虑到同一时刻有多个进程，很显然必须要有一种机制将进程地址空间映射到实际主存上。映射由虚存管理器（Virtual Memory Manager）来完成。“虚存”中的“虚”字表示进程存储的大部分（或全部）并不是实际主存，不在主存中的地址空间的内容存储在硬盘上，如图 1-2 所示。

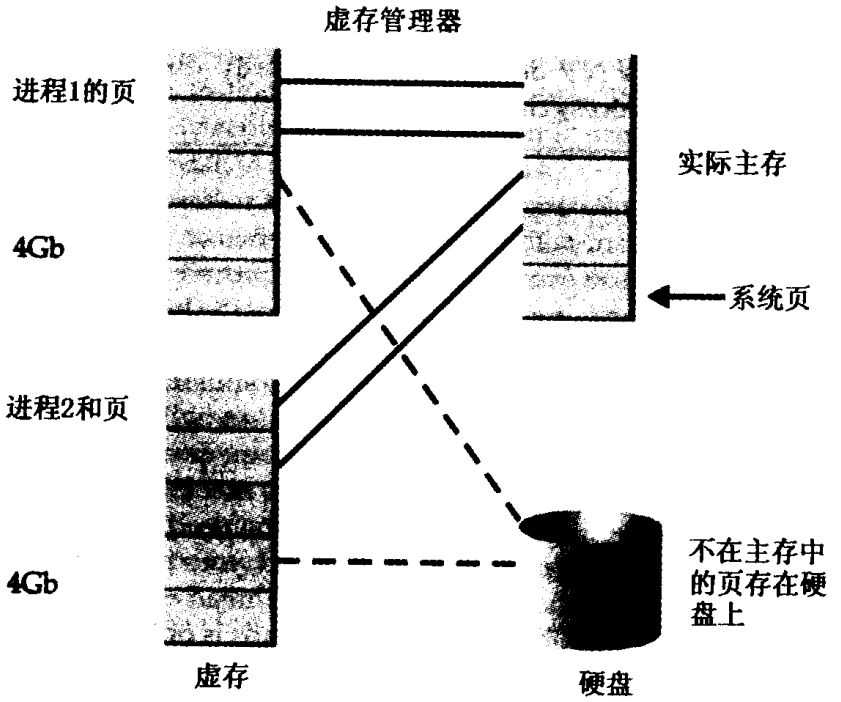

图 1-2　虚存管理

在执行期间，程序可能需要额外的存储，这可能是因为程序控制转移到了未驻留在主存的代码段，或要求更多的主存保存应用数据。不管是哪种原因，因为实际主存有限，实际主存中的某些内容必须被换出。交换存储内容的处理称为请求调页（demand paging）。页（page）是将被换入或换出的最小存储容量，典型的页尺寸为 4K。实际主存和进程地址空间都被划分成页，因此每个进程有（4Gb/ 4K）一百万页，若有 32MB 实际主存，则有（32MB/ 4K）8000 个实际主存页。选择移出页的技术称为先进先出（FIFO）。操作系统跟踪最先进入的页，并在换出时选择这些页。这种技术的思想是最近使用过的页有更大的机会被再次使用，因此要保存在实际主存中。虚存管理器使用页表（page table）（真正的页表是多级表）来保存所有主存页的状态信息。商业应用程序员通常不用考虑操作系统内部如何使用存储管理过程来管理存储。

虚存管理力求保证适当的平衡。若分配给进程的主存太多，则只能运行少量进程，且某些进程因占用的实际主存不能被快速访问而浪费了；分配给进程的主存太少将导致频繁地做页交换，使得操作系统占用了许多应用程序进程 CPU 时间（这种情况发生时，系统将是抖

动（thrashing）的）。由于不同进程访问存储的方式不同，对一进程为最优的分配可能对另一进程不是最优的，因此为了取中，操作系统将监测分配的页数和被每一进程使用的页数（也称为工作集（working set）），并自动地微调存储的分配。

Windows NT 使用 32 位线性存储地址。线性存储（linear memory）意味着整个存储被视为一个大的平面，每个地址是 32 位地址空间中的一个值。Windows 3.1 中的存储模式是分段存储，即存储被视为是由 64k 的段组成的。若您熟悉分段存储的编程，可将 4Gb 存储看作为一个巨大的段，不用再为远指针和近指针担心。4Gb 的限制是从 32 位地址（2^{32}）得来的。进程地址空间的一半用于应用程序进程，另一半用于系统功能（与应用程序进程相关）。

致 UNIX 程序员

UNIX 也使用线性存储编址，另外 Windows NT 和 UNIX 都使用请求调页，都支持存储映象文件（memory-mapped files），这是一种加速文件访问的技术，它将文件保存在主存中而不是硬盘上。另外两种系统都使用堆（heap），这是一种非结构化的存储。当然两者也有一点不同，NT 有更丰富的虚存和堆管理 API。

输入/输出管理器

NT 执行的这一部分处理所有的输入和输出，包括显示器、磁盘和 CD-ROM 等的输入输出。I/O 管理器使用统一驱动器模式（uniform driver model），这种模式中，对设备的第一次 I/O 请求都通过 I/O 请求包（I/O request packet，IRP）；而不考虑特定的 I/O 设备。设备细节由 I/O 管理器的下一层处理。I/O 管理器异步地执行 I/O 任务，发出 I/O 请求的进程被操作系统抢先，并等待直到收到 I/O 已完成的信号。

Windows NT 中的 I/O 管理器使用了大量的子构件，如网络重定向器/服务器（远程访问服务器，Remote Access Server 或 RAS）、Cache 管理器、文件系统、网络驱动器和设备驱动器：

■网络重定向器/服务器（RAS）　在第 3 章中讨论。

■Cache 管理器　在 NT 与 UNIX 中，Cache 用于存储磁盘上被频繁访问的数据（如文件数据）以加速对该数据的再次访问。与其它固定 Cache 尺寸的系统不同，NT 中的 Cache 尺寸可根据使用的存储数量不同而变化。

■设备驱动程序（Device drivers）　需要设备驱动程序的原因很简单，NT 可能要连接上百台打印机、磁盘驱动器、CD-ROM 驱动器和其它外部设备。驱动每个设备的底层代码只用于该设备。例如 HP 打印机的走纸命令可能与 Epson 打印机命令不同，甚至同一个厂家的不同型号的打印机命令也不同。若一字处理进程要打印，则要求 NT 基本操作系统针对特定设备格式化输出是无意义的，因此为特定设备格式化输出的任务由设备驱动程序完成，如图 1-3 所示。

致 UNIX 程序员

NT 和 UNIX 将所有的 I/O 数据视为字节串或文件，都实现抢先的 I/O 请求。

Windows NT 基本操作系统以标准方式与所有设备驱动程序交互。与此不同，Windows

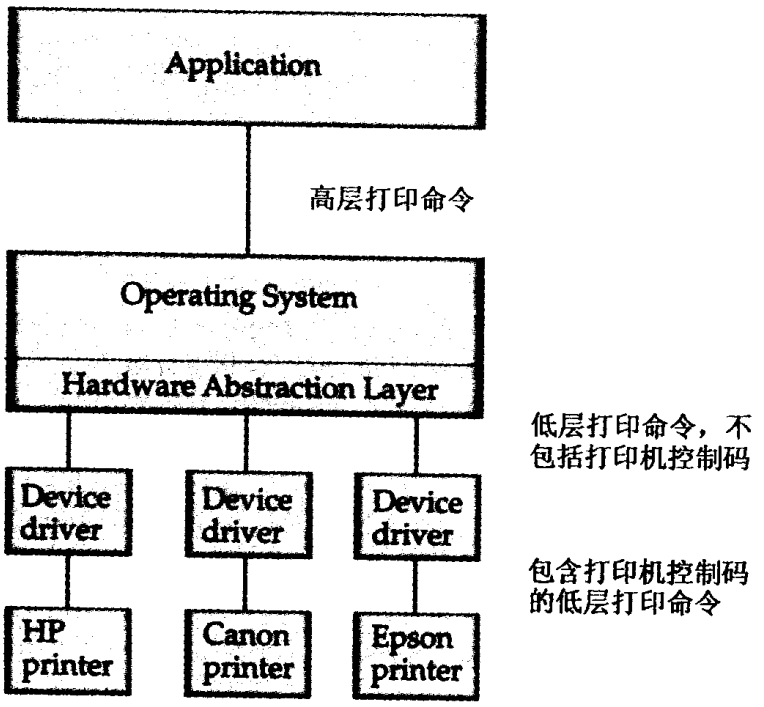

图 1-3　设备驱动程序增加了特定设备代码

95 支持两种方式——某设备驱动程序可为保护模式的驱动程序，或可为实模式驱动程序。如前所述，Windows NT 限制了底层硬件访问，不支持实模式驱动程序也是这一设计原则的体现。NT 包含了设备驱动程序工具包（device driver kits，DDK），与 SDK 一样用于编写设备驱动器。通常设备驱动程序由生产外部设备的厂家或专门的设备驱动程序软件公司编写，它也是商业应用程序员不用考虑的底层程序设计。

文件系统（file system）子构件处理对文件的访问，读和更新操作，将在第 4 章介绍。

对象管理器

执行的这一部分负责创建、管理和删除对象。Windows NT 操作系统中几乎每部分都是一个对象，例如，主存、进程、设备等都是对象。对象有与之关联的属性，对象功能的激活通过方法（methods）完成。从对象的创建到被删除期间，都有唯一的标识对象的句柄（handle）与之关联。

安全性引用监测器

由于黑客和病毒的存在，计算机系统的安全性正在引起人们的重视。随着因特网的普及，黑客和病毒的侵入机会急剧增加。任何有外部连接和（或）软驱的计算机都有被破坏和被病毒侵入的可能，因此操作系统增强了安全性功能。联邦政府的许多机构和计算机工业组织研究安全问题已有很长一段时间了，这些努力的结果之一是制定了操作系统应提供的安全性级别。级别从 D（最不安全级）到 A（最安全级），每级再细分。许多操作系统开发商的目标是 C2 级，这也是很多联邦计算机供应中要求的，NT 达到了所要求的 C2 安全级。

Windows NT 在所有资源中都实现了安全系统，例如文件有一个关联的安全级。操作系统的所有用户，包括系统管理员、其它用户和应用程序都有与其关联的安全级，安全信息用一个访问控制列表（access control list）描述。当某个用户要访问某一资源时，比较两者的安全级以判断该用户是否可访问这个资源。当然，安全性比这要复杂得多。安全性遍布于整个

Windows NT，并且是许多 API 和 MFC 类的一部分。关于 API 和 MFC 类的讨论贯穿全书，许多建立和维护安全性的函数由系统管理员使用。

本地过程调用机制

应用程序（客户，client）要向保护子系统（服务器，server）请求服务，例如，一个 Win32 应用程序从 Win32 子系统请求服务。尽管“客户/服务器”使人联想到一小型的 Windows 桌面系统通过网络访问大型服务器，但客户和服务器也可并存于同一台机器中。对分布于网络上的客户和服务器，最常用的通信机制是工业标准——远程过程调用（remote procedure call，RPC）。当客户与服务器并存于同一台机器中时，就有共同的资源如共享的存储，因此就有了优化通信机制的可能，这样优化为本地过程调用（local procedure call）。

1.1.4　保护子系统

不由内核执行的操作系统功能由一组非高优先级的服务器执行，称为保护子系统（Protected Subsystem）（图 1-1）。当您的应用程序做了一个 NT API 调用时，这些调用由保护子系统处理。保护子系统方法的一个优点是允许开发新的保护子系统模块，而不会影响基本操作系统和其它保护子系统，同理可加强保护子系统。例如，当微软要撤消对 OS/2 或 POSIX 应用程序的支持时，则所有要做的工作只是去掉这些保护子系统中的代码。尽管 Windows NT 设计为可运行 POSIX 和 OS/2 应用程序，但不能在 Windows 应用程序和 POSIX（或 OS/2）应用程序间传递数据和文件；也不能看到 POSIX 和 OS/2 应用程序的图形用户界面，虽然 Windows 就是以图形用户界面闻名的，而 POSIX 和 OS/2 只支持字符模式应用程序。这些都是因为 Win32 子系统是主子系统，支持文本、图形、网络和所有 NT 提供的功能的编程。而 POSIX 和 OS/2 子系统是“兼容模式”子系统，只支持文本，而没有图形和网络支持。图 1-1 中 POSIX 或 OS/2 系统与 Win32 子系统间的连线指出，对大多数应用程序调用，OS/2 和 POSIX 子系统确实调用了 Win32 子系统。微软的文档专门讨论了 Windows NT 处理 Win32 和 POSIX（或 OS/2）应用程序的差别。

微软白皮书《Microsoft Windows NT from a UNIX Point of view》写到：

“POSIX 和 OX/2 子系统为其应用程序提供了‘兼容模式’环境，当然，没有 Win32 子系统那么丰富的特点。”

一本 Windows NT 资源工具的技术性更强的说明书中写到：

“对本版本 Windows NT，POSIX 应用程序不能直接访问 Win32 子系统的任何设施和特征，如内存映射文件、网络、图形或动态数据交换。”

换一种说法，不能访问图形意味着 Windows 流行的图形用户界面不是 POSIX 应用程序固有的特征，您被限制于文本控制台应用程序。没有网络意味着没有 WinSock、点到点协议（Point-to-Point Protocol，PPP）等支持。您将看到通过提供 POSIX 支持，微软确保 Windows NT 可用于联邦和州的招标，因为它与 POSIX 处理兼容，同时又能鼓励用户切换到 Win32 子系统并发挥其优势。

除了保护子系统，Windows NT 还支持虚拟机（virtual machine）。例如，所有的 DOS 程序在虚拟 DOS 机（Virtual DOS Machine，VDM）中运行。使用同样的机制可以运行 16 位 Windows 应用程序（也称为 WOW，即在 Win32 上的 Windows），16 位 Windows 应用程序可

能产生的 GPF 并且关闭 Windows 3.1 的现象将不会在此出现，因为应用程序只影响虚拟机，而不是整个操作系统。这种防护的代价是某些 16 位应用程序（游戏、访问硬件的应用程序或通过非标准形式以增强性能的程序）不能在 NT 上运行。Windows NT 可同时运行多个虚拟机。

1.2 编程语言和开发环境

微软件提供了大量的语言编译器和集成开发环境以支持程序开发。语言支持包括通用编程语言如 Visual C++、Visual Basic、Fortran，以及专用语言如 HTML、Visual J++ 和 PERL（Practical Extraction and Report Language）。

超文本标记语言（HTML）用于创建 WWW 主页和支持嵌入多媒体内容如图形到 Web 主页中。HTML 也支持超链接（hyperlinks），用户可点击链接（通常与其它文本颜色不同）到达超链接指向的地址 URL（Universal Resource Localor，统一资源定位器）。

PERL 源于 UNIX，它是一种自由形式的解释性语言，用于扫描、从文本文件中取信息，并用抽取出的信息打印报表，Windows NT 支持 PERL 5。

Fortran 主要用于大多数的科学应用和某些工程应用，微软的 Fortran PowerStation 支持最新的 Fortran 标准—— Fortran 90。

Java 是由 Sun Microsystem 公司开发的语言，与传统编程语言有一个重要差别，即 Java 编译器不为专用环境生成可执行代码，它输出 Java 字节代码（Java byte codes）。这些字节代码由 Java 字节代码解释器解释，解释器可嵌入到许多应用程序中。最典型的嵌入 Java 解释器的应用是 WWW 浏览器。Visual J++ 是微软的 Java 版本，它可在一个集成开发环境中创建、编辑、编译和调试 Java 程序。

Visual Basic 通过使用预建的构件和对象提供了不用大量编程就能建立应用程序的能力。它有许多其它编程语言的特征，如分支、条件求值、赋值等。它使用类似于英语的构造语言而不是精简的构造编程语言。

记住，您不会厌倦任何一种编程语言。对大部分语言，您可将程序从一种语言修改为用另一种语言编写，所有主要的开发精力通常为使用不止一种语言。学习编程语言就像学开车，一旦您对路有了基本的感觉，就可相对容易地驾驶许多种车辆。一旦您学会了用一种语言编程，并理解了逻辑、算术等，在很短的学习周期之后就可用其它语言编程。

微软件提供了一种称为 Developer Studio 的集成程序开发环境，它支持 Visual C++ 和其它语言的开发，提供的功能包括

- 源代码管理
- 编辑和编译源程序
- 链接编辑和制作文件
- 项目管理
- 调试

微软件在 Developer Studio 中集成了与程序开发相关的新功能，本书的许多例子将用 C++ 编写，并由 Developer Studio 开发和测试。

其它的软件开发商也提供了开发环境，例如，Borland 公司生产语言编译器、编程环境

和工具，如 C++ 编译器、窗口对象库（OWL）等。

1.3　应用程序编程接口

Windows 应用程序接口从 Win16 —— Windows3.1 使用的 16 位 API 开始。Win32 是 NT 使用的 32 位版本。Win32s 是 Win32 API 的子集，它可被 16 位应用程序调用。Wing 是图形 API。尽管 16 位应用程序仍有巨大的潜力，但硬软件的优势使将来的大多数程序使用 Win32 API。本书重点在 Win32 API，通过查看文件名中的“32”，可分辨系统文件和 DLL 是 16 位的还是 32 位的。例如，Kemd32.dll、Gdi32.dll、Vser32.dll 和 Regedt32.exe 都是 32 位函数。

win32 API 分类如表 1-1。

表 1-1　Win32 API 分类和功能

分　类	功　能
控件	组合框，对话框，编辑框，题头，热键，图像列表，列表框，列表视窗，进度条，特征页，多功能编辑框，滚动条，状态条，工具条，工具盒，跟踪条，树视窗和 CP-down 控件 例如：CreateUpDownContol
图形（GDI）	高级图形函数，包括动画 例如，CAnimateCtrl
窗口管理	窗口，窗口类，窗口特性和窗口过程：光标，菜单，钩子，图标，多文档界面（MDI），键盘加速键，键盘输入，鼠标输入消息，消息队列，时钟和剪贴板 例如：InsertMenuItem
控制台	与外壳相关的函数 例如 TextOut
系统服务	访问文件和数据库，异常处理，网络传输，性能监测，进程，线程和安全性 例如 GetOpenfileName
网络	处理网络连接，管理网络配置，域管理，网络管理，远程访问服务和 Windows Socket 接口 例如：GetExtension Version
多媒体	音频，媒体控制，视频，游戏杆和其它专用输入/输出 例如：mciSendString

win32 扩展

微软扩展了 Win32 API，包括的支持有

■使用电话应用程序编程接口（Telephony API）开发集成电话功能的桌面应用程序。

■使用远程访问服务器（RAS）开发异步通信应用程序。

■使用数据访问对象（Data Access Objects，DAO）向应用程序提供数据库访问。

■使用 Exchange SDK 开发电子商业应用程序。

1.4　软件开发工具

微软提供了大量的软件开发包（SDK）。SDK 是自包含的包，包括说明、程序员参考文档、源代码例子等。每个 SDK 集中于一个主题。Window NT 提供了大量 SDK，包括：

■Win32 SDK 它使您可使用 Win32 API 开发程序（参见 1.3 节）。

■MAPI SDK 它使您能开发 MAPI 应用程序。MAPI SDK 有 16 位和 32 位版，32 位版本是 Win32 SDK 的一部分，包含有 C、C++和 Visual Basic 例子程序。

■OLE SDK 它包含有 COM 说明，可使您使用微软的对象链接与嵌入接口开发应用程序。

■ODBC SDK2.1 它使您能开发开放数据库互连（ODBC）驱动程序和使用 ODBC 的应用程序。

■DAO SDK 可使您开发 DAO 应用程序。

■RAS SDK 可使您使用 RAS API 开发应用程序，能使用 RAS API 编写应用程序来与远程机器建立连接、通信和终止连接。

■Exchange SDK 它可使您能开发基于微软 Exchange 服务器的客户服务器程序。

■SMS SDK 它用于开发系统管理服务器应用程序。SMS 服务器和 SNA 服务器是附加的产品，它们在 Windows NT 之上运行。相关内容在本章后面讨论。

■系统网络结构（SNA）SDK 它用于开发 SNA 服务器应用程序。

■活跃目录服务器接口（ADSI）SDK 它可使用户（如管理员）或应用程序访问和操纵 LDAP、NetWare 和 NT 目录。ADSI 还包括话跃数据对象（ADO）和 OLEDB 接口，它本身不是语言，可与 Visual Basic、Visual J++和 Visual C/C++一起工作。

1.5 微软基础类库（MFC）

微软基础类库（MFC）是使用 Win32 API 开发应用程序的一个应用程序框架，是 C++类的集合，类简化了编程并促进了代码重用，使用 MFC 类节省大量时间。MFC 集成在 Visual C++开发环境中。例如，若使用 AppWizard 开发应用程序，您可获得要使用的 MFC 类。MFC 建于 Win32 API 之上，并试图将程序员与 API 编程的细节隔离开。除非有专用平台限制，用 MFC 开发的程序对不同的 Windows 平台有良好的移植性。如果您有一些老的 16 位代码，微软有一个 MFC 移植工具帮助您移植老的 16 位代码。MFC 紧跟微软已达到的开发技术，最新版的 MFC 包括了全部的 OLE 支持、DAO 支持、公用控件支持和线程同步支持等。

本书将包含大量的用 MFC 编写的例子和练习。

1.6 图形设备接口（GDI）

图形设备接口（graphical device interface）处理那些使 Windows 和微软闻名的图形终端用户界面（与基于字符的界面相比）。Windows 包含了大量开发 GDI 应用程序的支持，包括 Win32 API、MFC 类和预建控件等。您将用到的 GDI 对象有：

■画笔

■画刷

■字体

■位图

■元文件

第 2 章将详细介绍用户界面编程。